UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/759,697	01/12/2001	Robert H. Halstead, JR.	09612.1028-01000	2648

22852	7590	03/20/2006

FINNEGAN, HENDERSON, FARABOW, GARRETT & DUNNER LLP
901 NEW YORK AVENUE, NW
WASHINGTON, DC 20001-4413

EXAMINER
STEELMAN, MARY J

ART UNIT	PAPER NUMBER
2191	

DATE MAILED: 03/20/2006

Please find below and/or attached an Office communication concerning this application or proceeding.

	Application No.	Applicant(s)
	09/759,697	HALSTEAD, ET AL.
Office Action Summary	Examiner	Art Unit
	Mary J. Steelman	2191

-- *The MAILING DATE of this communication appears on the cover sheet with the correspondence address* --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE <u>3</u> MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133).
  Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *12/23/2005, 3/9/2005.*

2a)☐ This action is **FINAL.**   2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle,* 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-27* is/are pending in the application.

    4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-27* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

    Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

    Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

    a)☐ All b)☐ Some * c)☐ None of:

      1.☐ Certified copies of the priority documents have been received.

      2.☐ Certified copies of the priority documents have been received in Application No. _____.

      3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

    * See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1)☒ Notice of References Cited (PTO-892)

2)☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)

3)☐ Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08)
    Paper No(s)/Mail Date _____.

4)☐ Interview Summary (PTO-413)
    Paper No(s)/Mail Date _____.

5)☐ Notice of Informal Patent Application (PTO-152)

6)☒ Other: *See Continuation Sheet.*

Continuation of Attachment(s) 6). Other: Accepted Replacement Sheet Fig 2F.

DETAILED ACTION

1.   This Office Action is in response to Remarks received 12/23/2005, and Amendments and

Remarks received 3/9/2006. Per Applicant's request, claims 1, 1, 13, 25, and 26 are amended.

Claim 28 is previously cancelled. Claims 1-27 are pending. Per Applicant's request the

Specification has been amended. Replacement Sheet Drawing 2F has been received. A 37 CFR

1.132 Declaration from Dr. David A. Kranz has been received.

Double Patenting

2.   The nonstatutory double patenting rejection is based on a judicially created doctrine

grounded in public policy (a policy reflected in the statute) so as to prevent the unjustified or

improper timewise extension of the "right to exclude" granted by a patent and to prevent possible

harassment by multiple assignees. A nonstatutory obviousness-type double patenting rejection is

appropriate where the conflicting claims are not identical, but at least one examined application

claim is not patentably distinct from the reference claim(s) because the examined application

claim is either anticipated by, or would have been obvious over, the reference claim(s). See, e.g.,

In re Berg, 140 F.3d 1428, 46 USPQ2d 1226 (Fed. Cir. 1998); In re Goodman, 11 F.3d 1046, 29

USPQ2d 2010 (Fed. Cir. 1993); In re Longi, 759 F.2d 887, 225 USPQ 645 (Fed. Cir. 1985); In re

Van Ornum, 686 F.2d 937, 214 USPQ 761 (CCPA 1982); In re Vogel, 422 F.2d 438, 164 USPQ

619 (CCPA 1970); and In re Thorington, 418 F.2d 528, 163 USPQ 644 (CCPA 1969).

A timely filed terminal disclaimer in compliance with 37 CFR 1.321(c) or 1.321(d) may

be used to overcome an actual or provisional rejection based on a nonstatutory double patenting

ground provided the conflicting application or patent either is shown to be commonly owned

with this application, or claims an invention made as a result of activities undertaken within the

scope of a joint research agreement.

Effective January 1, 1994, a registered attorney or agent of record may sign a terminal

disclaimer. A terminal disclaimer signed by the assignee must fully comply with 37 CFR

3.73(b).

3.   Claims 1-27 are provisionally rejected on the ground of nonstatutory obviousness-type

double patenting as being unpatentable over claims 1-25 of copending Application No. 09 /

760031. Although the conflicting claims are not identical, they are not patentably distinct from

each other because they are drawn to substantially the same invention of defining an object with

an option data structure which supports references to option values without preallocation of

memory space.

This is a provisional obviousness-type double patenting rejection because the conflicting

claims have not in fact been patented.

Response to Arguments

4.   Applicant's arguments with respect to claims 1-27 have been considered but are moot in

view of the new ground(s) of rejection.

It appears to Examiner that Applicant is defining an object oriented language, using a class

template type of structure which includes optional elements (option data structure) stored in a

linked list or another type of data structure. To save memory, memory is dynamically allocated

at compilation, as needed. At compilation, when a get_selected_option_value() or

set_selected_option_value() type of method is encountered, type checking is done using the

'option data structure'. Examiner contends that dynamic allocation of memory in an object

oriented environment, using type checking is well known in the art.

Claim Rejections - 35 USC § 101

5.   In view of the cancellation of claim 28, the prior 35 USC 101 rejection is hereby

withdrawn.

Claim Rejections - 35 USC § 102

(e) the invention was described in (1) an application for patent, published under section 122(b), by another filed
in the United States before the invention by the applicant for patent or (2) a patent granted on an application for
patent by another filed in the United States before the invention by the applicant for patent, except that an
international application filed under the treaty defined in section 351(a) shall have the effects for purposes of this
subsection of an application filed in the United States only if the international application designated the United
States and was published under Article 21(2) of such treaty in the English language.

6.   Claims 1-27 are rejected under 35 U.S.C. 102(e) as being anticipated by USPN 5,999,948

to Nelson et al.

Per claims 1, 13, 25, and 26:

-defining a class which supports an option data structure having, in instances of the class,

references to option values without preallocation of memory space for the full option values, the

option data structure including a type description of the option values;

Nelson: Col. 5, line 48-49, "generates the necessary Factory objects and Form specification classes (defining a class)." Col. 5, lines 39-41, "Then the form section (option data structure) is created grouping and identifying the various widgets (without preallocation of memory space for the full option values, only selected widgets are references). Col. 4, lines 61-67, "The form engine 19 then collects the corresponding data and reads any layout information from the corresponding FDL file...The dynamic form 1 is then presented..." Col. 8, lines 13-15, "The Dynamic Forms tool has the unique ability to automatically select a widget type (type description) based on the characteristics of the data item being displayed by the widget (the selected option value)."

Col. 7, 14-31, "The process of dynamically creating both a dynamic form and a Data Collection at run-time based on a form description is complex...The design of dynamic forms isolates the object creation functions within 'factory' objects using the abstract factory pattern description...These factory objects do nothing more than create instances of specific types of objects, such as a Windows Combo Box widget (option value). The factory objects are controlled by 'factory manager' or builder objects the (builder) objects are responsible for interpreting the data from the form description, and invoking the factory objects to create the appropriate objects.

-compiling an operation on an option value in an instance of the class using the type description in the option data structure.

Nelson: Col. 12, lines 61-67, "During the exchange, the DataItem for each widget is looked up in the DataCollection and the appropriate read/write function is executed. A read gets data (get() operation compiled on an option value in an instance of the class) from the DataItem and places it in the widget (class instance of an option value, using the type description in the option data structure) and a write sets data from the widget into the DataItem. The DataItems maintain all the necessary information for each particular piece of data. DataItems maintain type information...".

Per claims 2 and 14 :

-the option data structure identifies change handler code that is executed when an option value changes.

Nelson: See FIGs. 3A and 3B. The value of the widget is changed, #25 & #29. Related text found at col. 5, lines 15-23.

Per claims 3 & 15:

-change handler code for one option is defined in different classes with a class inheritance hierarchy and the change handler code from each class is executed when the option value changes.

Nelson: Col. 8, lines 6-7, "each form description is translated into JAVA source code, this can then be compiled into JAVA class files (code defined in classes with class inheritance hierarchy, inherently when using JAVA programming language) Col. 3, lines 25-37 disclose a dynamic form software engine that retrieves an FDL file for the corresponding configuration form. The FDL file determines the layout type. Appearance (col. 3, lines 40-45) is determined by a policy set in the forms engine and also determines the type of widget presented. Col. 3, lines 45-51, "When new configuration forms are needed, or old configuration forms needed to be revised, only new FDL files need to be released and added to existing applications...to change the style of a configuration form, the policy of the form engine only needs to be changed

(execute change handler code)." FDL files are registered with the form engine (col. 5, line 3) and added to an internal catalogue (col. 5, line 9). Col. 6, lines 51-58, "easy to add new types of widgets (options) and forms...new types of DataCollection...new types of data bases storing data in a different form or new protocols for conveying the data..." Col. 7, lines 46-53, "First it becomes very easy to upgrade the application to include new forms to support new hardware devices. New FDL files are copied into the applications run-time directory...configuration forms for new modules can be added while the application is running." The forms engine handles 'change handler code' for changing option values.

Per claims 4 & 16:

-wherein the option data structure includes a default value...

-in a get operation to an instance of the class, if an option value which applies to the instances has been set, getting the set option value and, if no value which applies has been set, getting the default value for the class.

Nelson: Col. 5, lines 10-23, "The form engine 10, or FDL files 13 can contain rule descriptions which also control values of individual items of the data....include limiting the maximum and minimum values (default values)...and also adjusting the valued of a second item, based on a new value of a first item (default values). Additional discussion related to get() and set() operations is found at col. 5, lines 15-23 and FIGs. 3A & 3B. Col. 5, lines 27-33, "The form construction process includes the selection of widgets, the binding of each widget to a data source (default value or set value)...and the construction of rule objects that respond to the users interaction (user setting a value) with the form and validate the users input."

Per Claims 5 & 17:

-defining a first class with a first option data structure of a first form which supports, in instances of the class, references to option values without preallocation of memory space for the full option values

See rejection of limitations in claim 1 above. Col. 5, line 25, "The form engine finds the entry for the form in the catalogue, constructs the form using the information from the

specification objects, and displays..." Memory is not allocated for forms in the catalogue that are not used for a particular request / display.

-defining a second class with a second option data structure of a second form which supports, in instances of the second class, references to option values without preallocation of memory space for the full option values, the second form being different from the first form

As an example, a different form may be chosen from the catalogue to define a second class with a second option data structure of a second form.

-during compilation, encoding an option operation as a method call to an object of the first class and to an object of the second class without regard to the form of the option data structure supported by the class.'

Nelson disclosed (col. 3, lines 17-24) "a form description language (FDL) for describing basic elements of a configuration file...includes a description of the data which is presented and is modifiable by the form...the FDL file can also contain a basic layout type for the data." Col. 3, lines 38-51, "Variations in the appearance of a configuration form, such as size, spacing and type of components is determined by a policy set in the forms engine...certain types of data will always have a certain type of widget presenting and modifying the data...When new configuration forms are needed (defining a second class with a second option data structure)...only new FDL files need to be released and added to existing applications...when it is desired to change the style of a configuration form, the policy of the form engine only needs to be changed..."

Nelson allowed for multiple configuration forms (first, second), using appropriate FDL files. Col. 4, lines 22-39, "Inside the application 9 are a plurality of FDL files 13. These contain a description of the different types of data 7 in the form, and more specifically includes specification for obtaining the data 7, i.e., descriptions of MIB (Col. 1, line 45 / Management Information Base)...The FDL files 13 while part of the application 9, are separately insertable and removable into and out of the application 9. When the application 9 desires to resent a

dynamic form.1, it sends a form type request 15 and a data.location request 17 to the form engine 19. The form engine then reads the FDL file 13 corresponding to the form type 15. The form engine 19 then creates the different types of DataCollections 21 for the appropriate data described in the corresponding FDL file 13."

Col. 4, lines 40-49, "The form engine 19 has a separate data collection portion 21 for each type of device and connection convention...In this way each form does not need to have its own code for retrieving and modifying data (without regard to the form of the option data structure supported by the class) in each device (option operation as a method call to an object of the first class and to an object of the second class)."

Per claims 6 & 18:

-notifying objects of a change in an option value through a change handler identified by an option binding, the option binding being located by first searching a mapping data structure for a previously computed mapping to the option binding and, if no mapping was previously computed, by then computing the mapping to the option binding and storing the mapping in the mapping data structure.

As an example, see FIGs. 3A & 3B. The objects have values bound to the option values. A change handler, in this example, has changed the State, #25 & the Redundant port, #29. The previously computed mapping for state was 'enabled.' The computed / stored mapping was changed to 'redundant-primary.'

Per claims 7, 19, and 27:

-the option data structure comprises a linked list of option items having option values.

Nelson: Col. 5, lines 56-62, "The Dynamic Forms engine constructs the form using the factories and specification in the catalogue...The factories create the elements of the form...The factories layout all the elements in the form based on the Dynamic Forms Engine's layout policy. Col. 6, lines 19-20, "...software package containing C++software classes for Lists, Strings, and other generic tools that are used. Col. 7, lines 17-24, "The design of dynamic forms isolates the object creation functions within 'factory' objects using the abstract factory pattern description as described on page 87 in the reference, Design Patterns Elements of Reusable Object Oriented

Software', by Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides..." The

Dynamic Forms engine uses factories and the specification in the catalogue. The catalogue

contains option items. Data sources fill the values.

Per Claims 8 & 20:

-a nonlocal option value applies to other objects in a nonlocal option hierarchy.

Nelson disclosed a global description (non local option hierarchy) of layout at col. 3, ·

lines 17-18: "a form description language (FDL) for describing basic elements of a

configuration file." Col. 3, lines 29-37, "The form engine retrieves the FDL file for the

corresponding configuration form. The form engine reads the respective FDL file, and parses the

data described by the respective DFL file. The form engine then creates the secondary window

in the application with a layout in accordance with the layout type

Per Claims 9 & 21:

-the nonlocal option hierarchy is a graphical hierarchy.

Nelson disclosed a graphical programming application. Col. 4, lines 19-20, "The widgets

5 display data 7, and this data can be changed by the operator of the application, usually through

the widget." Col. 4, lines 63-66, "The policy portion 23, of the form engine 19, "arranges the

data according to any layout information in the FDL file 13, and adds additional appearance and

style features to create the final appearance of the dynamic form 1." Col. 6, lines 1-25 describe

an object oriented type of environment, which inherently is a hierarchical arrangement. A

nonlocal (global) option hierarchy (col. 5, lines 2-9) is provided by registered FDL files ...and

are added to the internal catalogue (nonlocal option hierarchy) of compiled form descriptions."

Col. 5, lines 47-49, "The application using dynamic forms registers the FDL file with the

Dynamic Forms Engine ...The Dynamic Forms engine parses the grammar in the FDL file and

generates the necessary Factory objects and Form specification classes (hierarchy)." Col. 5,

line 64, "The completed form is displayed on the screen (graphical hierarchy)."

Per Claims 10 & 22:

-the class which supports the option data structure includes defined fields to support

values in preallocated memory space.

Nelson: Col. 5, lines 55-62, "The Dynamic Forms engine constructs the form using the

factories and specifications in the catalogue...The factories create the elements (defined fields to

support values in preallocated memory space) of the form."

Per Claims 11, 12 23, and 24:

-the type description is used to check the declared type of a value to be set in a set

operation.

-the type description is used to check the legality of an operation to be performed on a

value obtained in a get operation.

Nelson: Col. 5, lines 10-12, "The form engine 19, or FDL files 13 can contain rule

descriptions which also control values of individual items of the data." Col. 6, lines 15-16, "All

accesses to the data (get / set operations) are handled by the interface to the DataCollection."

Col. 6, lines 66-67, "the form engine has the ability to retrieve (get operation), validate and store

(set operation) the data." See discussion on "get" and "set" operations at col. 12, line 28 through

col. 13, line 37. Col. 12, lines 61-62, The DataCollection contains...DataItems. Col. 13, lines 3-

8, "The DataItems maintain all the necessary information for each particular piece of data.

DataItems maintain type information...DataItems also contain any constraints..." Col. 13, lines

33-37, "Implicit constraints are part of the basic DataItem types...Each DataItem type has built

in rules for its type..." Col. 13, lines 19-22, Constraints are used by the designer of a form to

enforce limitations (checks the legality) on the values of items in a form. An error string is

specified with each constraint and this string is displayed when a constraint detects an invalid

value."

Conclusion

7.   The prior art made of record and not relied upon is considered pertinent to applicant's

disclosure.

Note pertinent references:

USPN 6,223,342 B1 to George (1996) A system and method for data processing objects

having a number of attributes (Abstract). Col. 2, lines 50-53, "Preferably, the global

configuration stream includes a count of attributes of the object, the order of the sequence, and

the individual attribute details, such as data types..." Col. 3, lines 12-21, "For an object having

three attributes...an object list includes the associated local configuration stream and three smart

pointers to memory locations (references to option values) of the values for the three

attributes...Within a sequence of object lists, some to the objects will be 'sparse,' i.e. some of the

objects will have one or more absent attributes (reference to option values)." Inherently

'compiling an operation on an option value in an instance of the class using the type description'

is analogous to George's system and method for data processing objects, having an optional

number of attributes.

USPN 6,842,906 B1 to Bowman-Amuah DETX (2794)

Detailed Description Text - DETX (2794):   "Make use of "lazy" or "deferred" loading. That

is, don't do a "deep" instantiation until you know you're going to use the associated parts of the

object. Instead, load selected sub-objects only when first referenced. This can save on memory

overhead as well as DBMS access. In some cases you can use a hybrid strategy: do a "shallow"

instantiation by default, but provide the client program with a way to build the complete object

on demand to provide more deterministic performance. One thing to be careful of with this

approach is that if you really do tend to use most parts of the object during high-volume

processing, loading it in piecemeal can actually worsen the performance, because of the

overhead of maintaining the load state and because of the smaller DBMS transactions sizes.

These techniques have a very small impact on your object model."

A shortened statutory period for reply to this final action is set to expire THREE

MONTHS from the mailing date of this action. In the event a first reply is filed within TWO

MONTHS of the mailing date of this final action and the advisory action is not mailed until after

the end of the THREE-MONTH shortened statutory period, then the shortened statutory period

will expire on the date the advisory action is mailed, and any extension fee pursuant to 37

CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event,

however, will the statutory period for reply expire later than SIX MONTHS from the date of this

final action.

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Mary Steelman, whose telephone number is (571) 272-3704. The examiner can normally be reached Monday through Thursday, from 7:00 AM to 5:30 PM If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Wei Zhen can be reached at (571) 272-3708. The fax phone number for the organization where this application or proceeding is assigned: 571-273-8300.

Any inquiry of a general nature or relating to the status of this application should be directed to the TC 2100 Group receptionist: 571-272-2100.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).
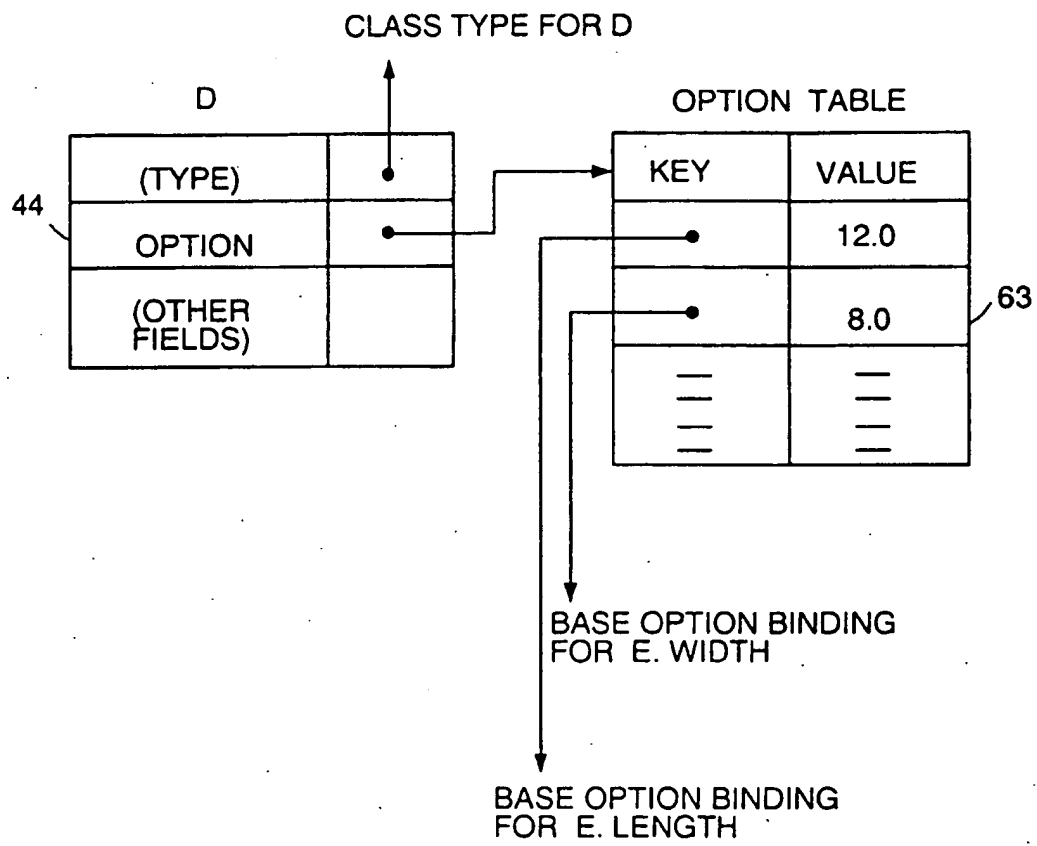
Mary Steelman

03/15/2006

CLASS TYPE FOR D

D

OPTION TABLE

(TYPE)		KEY	VALUE
OPTION			12.0
(OTHER FIELDS)			8.0

44

63

BASE OPTION BINDING
FOR E. WIDTH

BASE OPTION BINDING
FOR E. LENGTH

FIG. 2F